

LIBRARY OF THE
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

510.84

Il6r

no. 463-468

cop. 2



Math

160

0.465

GRASS: SYSTEM OVERVIEW

by

M. J. Michel

July 1971



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

REPORT NO. 465

GRASS: SYSTEM OVERVIEW*

by

M. J. Michel

July 1971

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS
URBANA, ILLINOIS 61801

* This paper supported in part by the Atomic Energy Commission under grant U. S. AEC AT(11-1)1469.



Digitized by the Internet Archive
in 2013

<http://archive.org/details/grasssystemoverv465mich>

ACKNOWLEDGMENT

The help of the following individuals who worked on segments of the system is greatly appreciated. The buffering control interface (PDP-8 to PDP-8/I link) and the satellite filing system were designed and implemented by Mr. Harold Levin. The buffering control (PDP-8/I software) was provided by Mr. Roger Haskin. Mr. Charles Hyde provided the display terminal multiplexer and the disk interface, while the remote computer filing system was implemented by Mr. Alfred Whaley. The general advice of Professor C. W. Gear was also of great benefit.

PREFACE

A relatively low cost, stand-alone, timeshared, interactive graphics system implemented at the University of Illinois is described. The facility, using a mini-computer, storage CRT terminals and a disk, supports eight consoles with a two-dimensional drawing package, text editor, full subpicture capability, and individual user libraries. The software architecture allows other graphics applications to be added to the system by means of transient program segments. The complete system can be reconstructed with currently available, off-the-shelf, hardware. Communication is also provided to a large, remote computer. Several user written applications programs can execute concurrently under a timesharing monitor that runs within the remote computer's standard operating system. These user programs can communicate with programs in the mini-computer, with users at terminals directly, and with an archival filing package in the remote computer.

TABLE OF CONTENTS

	<u>Page</u>
ACKNOWLEDGMENT	
PREFACE	
1. INTRODUCTION.	1
2. CONFIGURATION: CHOICES AND CONSIDERATIONS.	6
2.1 <u>Satellite Section</u>	6
2.2 <u>Remote Section</u>	9
3. SOFTWARE ARCHITECTURE: A USER'S VIEWPOINT.	12
4. OPERATIONAL EXPERIENCE.	18
5. EXTENSIONS.	20
6. CONCLUSIONS	22
LIST OF REFERENCES.	23
APPENDIX.	25

1. INTRODUCTION

As part of a larger investigation into a general simulation and modeling system, a relatively low-cost, stand-alone, timeshared, interactive graphics facility has been developed at the Department of Computer Science, University of Illinois at Urbana-Champaign. The purpose of this paper is to provide an overview of the facility. Detailed information can be found in the references cited.

The utility and advantages of interactive online graphics terminals have been well demonstrated over the past several years. Beginning with Sutherland's classic "SKETCHPAD" in 1963, graphics terminals and their applications as flexible man-machine interfaces have been steadily proliferating. However, the "graphics" terminal most often seen even today is merely a keyboard with an undersized alphanumeric display area. Even though some of these devices can perform a few elementary text operations on data in local character buffers, they hardly qualify as general graphics terminals. With the bare minimum price starting at three times the cost of a Teletype, the major benefits that these devices have to offer are silent operation and a slight speed increase. General graphics terminals for online use are few and far between; in fact, they are found in regular use only in a small number of large commercial companies, government agencies, and university research centers. The main reason is, of course, the high cost involved.

Caveat emptor is the rule. No matter what the friendly terminal salesman might say, something (particularly terminal hardware) is definitely not delivered for nothing. With this in mind, state-of-the-art hardware for color displays, three-dimensional perspective rotations and translations, hidden line removal, windowing, and shading is priced quite beyond the reach of small, medium, and for that matter, even most large installations. Simulating these facilities in real time for productive (as opposed to demonstration) purposes with software is also well known to require a very large investment in processing power, even for just a single terminal. Technology cannot quite support interactive computer graphics that appears to the terminal user as three-dimensional, color motion pictures, all at a

price comparable to that of seeing a John Wayne movie (e.g. \$1/hr.). However, some graphics facilities are economically feasible, and their power is considerable, particularly in light of how much can be accomplished with the primitive capabilities of a card reader and impact printer.

For ease of discussion, hardware/software systems that support the features mentioned previously (e.g., three-dimensions, windowing, etc.) will be termed simply "3D." The term "2D" will apply to systems that support comprehensive two-dimensional graphics capabilities. These capabilities include line drawing (with instancing and connection abilities), sophisticated text manipulation, maintenance of archival user libraries, and flexibility of application. Although a continuum of systems exists between 2D and 3D, the general assumption being made is that 3D systems tend to be too expensive to allow their wide use in the computing community. On the other hand, a carefully configured 2D system would be widely available, e.g., economically feasible. Another assumption functioning as a guideline is that using a less expensive, less intelligent terminal implies using more computer resources to support it; conversely, the more expensive, more intelligent terminals require less computer support. Dedicating a computer to a few unintelligent terminals is expensive, but using intelligent terminals is itself expensive. Even compromising is not effective: try moderately intelligent terminals connected to a computer capable of handling background batch stream or standard keyboard terminals. Such a configuration tends to reduce the superficial cost of the graphics support, but often results in an unpleasant degradation in system (both batch and terminal) performance. Discrete response time at the terminals may be long, and certain resources, such as computer main memory allocated to the graphics terminals, remain idle much of the time, so reducing batch or keyboard capabilities.

A typical 3D system would include at most two or three terminals directly attached to a large computing facility containing extensive peripheral resources. These terminals cost anywhere from \$50K to \$200K

each, depending on the nature of the hardware involved and the 3D features supported. Trade-offs involve such things as the method of display refresh (e.g., individual memories, shared memory, computer main memory, drum, or disk) and the "intelligence" of the terminal (e.g., what the terminal can do on its own: display only, data structure manipulation, or general stand-alone processing). Examples of terminals suitable for the above systems include the IBM 2250, the DEC 338, and various terminals made by ADAGE; the intelligence of these terminals runs from low to high, respectively.

Typical 2D systems can look very similar to the above 3D systems. However, the peak intelligence of the terminal and the maximum computer support will be much lower. An example of a minimal terminal consists of a display with a very simple controller refreshing out of its own memory. This type of device would cost about \$20K with half of that going for an adequate size memory. Of course, the terminal is completely dependent on support from the computer. If the computer dedicates enough support to the terminal, a system fairly close to a low level 3D could be reached. Such support would be expensive in terms of the cost of the dedicated resources. On the other hand, a lower level of support implies an increase in the number of terminals serviced or an increase in computer resources available for other uses. Upgrading the intelligence of a terminal in a 2D system, even by small increments, rapidly decreases the amount of computer support needed. An example of the extreme is provided by a configuration that has just recently appeared: a display serviced by its own (dedicated) mini-computer and a small disk. In this case, ALL 2D functions can be completely supported locally; support from the main computer is needed ONLY for (1) permanent library maintenance, and (2) analysis number-crunching. Hardware for such a 2D terminal would list at about \$40K. The cost is still far above a "drop in the bucket," but the difference between this very intelligent 2D terminal and an unintelligent 3D terminal in terms of both terminal hardware price and main computer support, is quite significant.

Yet further reduction in cost without facility or performance loss is still possible. Note that the mini-computer and disk at each terminal is obviously idle during user "think time," e.g., most of the time. The tempting improvement is to service several terminals with only one mini-computer and one disk. Carefully assessing the requirements of a 2D system and careful planning does indeed allow this to be accomplished. The result is a multi-terminal, flexible, and comprehensive 2D drawing system, entirely supported on a local mini-computer with access to a remote computer, only for occasional, very limited support. The average retail off-the-shelf per terminal cost for this system is only \$20K; the cost of an eight terminal configuration would be about \$160K or less than twice the cost of a single unsupported 2250 Model 1!

Three typical applications that this 2D system can support are graphical design, document preparation, and network analysis. The first application includes such things as architectural plans, mechanical drawings, and magazine page layouts. The second includes the obvious: reports, articles, manuscripts, etc. The last encompasses all phases of problem definition, from defining the primitive elements through specifying constraining network equations. The remote computer facility needs to be called only when finished items are ready for archival storage or for numerical analysis of a completely defined network. All other work, and the complete job in the case of the first two applications, is entirely supported on the satellite mini-computer complex. An example of such a configuration, the Graphical Remote Access Support System (e.g., GRASS), is described in the following sections (c.f. Figure 1).

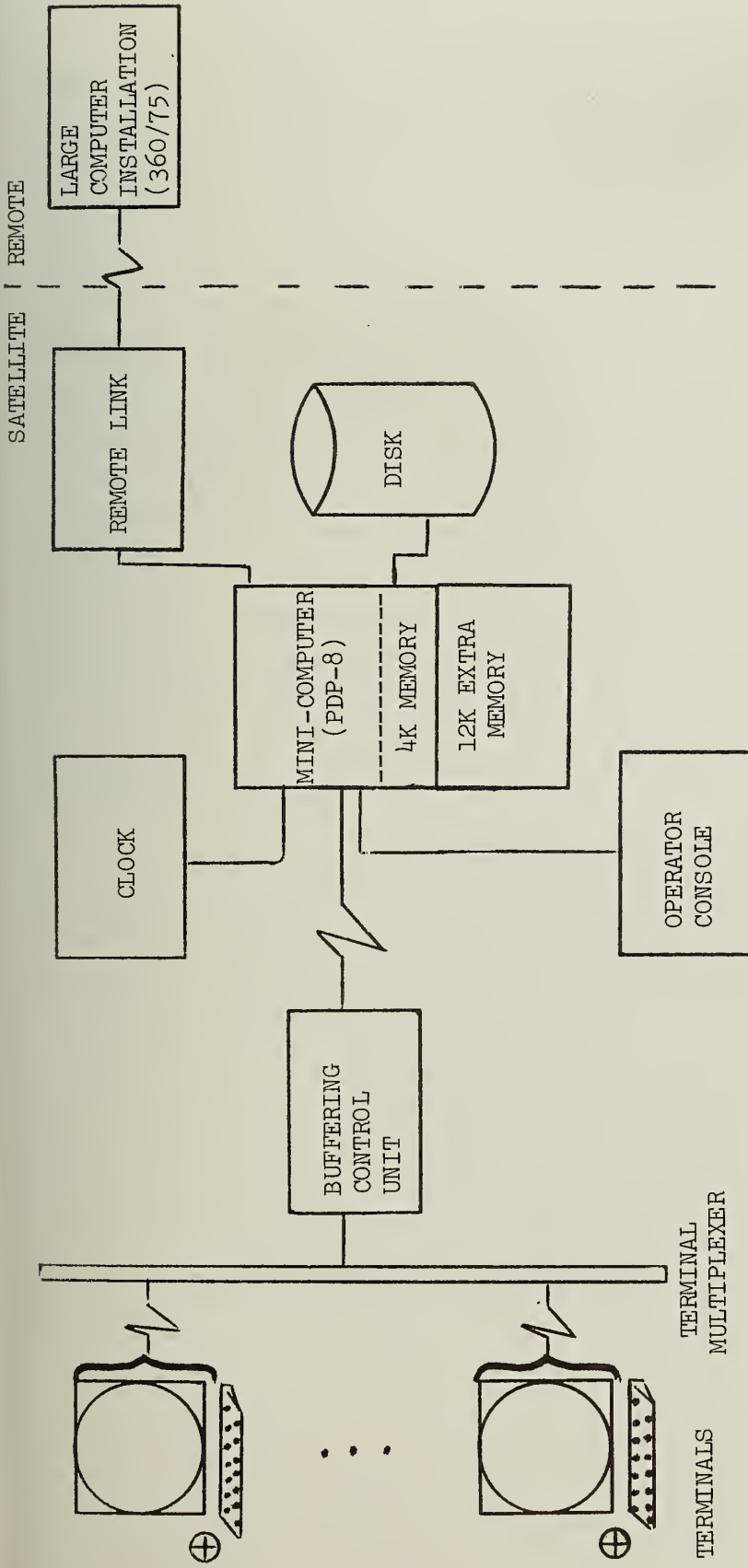


Figure 1. GRASS Hardware Configuration

2. CONFIGURATION: CHOICES AND CONSIDERATIONS

The configuration outlined in Figure 1 has been implemented in GRASS with the following items. The local (e.g., satellite) hardware includes eight direct view storage CRT terminals, each with keyboard and joystick, connected to a terminal controller, which is in turn linked to a PDP-8 mini-computer. This computer is responsible for the manipulation and temporary storage of all data structures and display files needed by the terminals. The terminal controller acts as a buffer and multiplexor for display files being shipped to the terminals and for keyboard lines and joystick (X, Y) input being shipped to the PDP-8. In addition, the PDP-8 has an interface for communication with a large remote computer, specifically, a 360/75. This remote computer is responsible for providing number-crunching and archival file storage. A graphics time-sharing monitor runs under the normal operating system of the 360/75; various user written applications programs can run concurrently under this monitor, communicating with programs in the PDP-8, with the terminal users directly, and with a 360/75 graphics filing system.

2.1 Satellite Section

The desire to implement a 2D multi-terminal graphics system using a single mini-computer immediately brings two problems to the fore. While "mini-computer" implies limited instruction repertoire, speed, and memory, "multi-terminal" implies a high interrupt-load environment. Hence, two critical features of the local portion of the configuration are its attempted maximization of the PDP-8's available library space, executable code, and buffering areas and minimization of its interrupt load. The aim of the first is to increase the capabilities available to the terminal user, while the intention of the second is to facilitate the concurrent support of several terminals. Inherent in both is that response time for the capabilities supplied be reasonable, and that total system cost be minimal.

Although the PDP-8 used is a very small, standard mini-computer with only seven instructions, two features were added to help make it uniquely powerful. First, extra memory was supplied for a total of 16K of twelve bit words. All of the memory used is standard speed core for this machine and, in fact, PDP-8's were designed to accommodate as much as 32K of this type of core. Unfortunately, very few installations have ever taken advantage of this feature of the machine. Second, a very high speed, high density head-per-track disk was added as a local mass store. Average latency is only 16.5 ms, the transfer rate is approximately one word every 5 μ s, and total space available is about 300K words. Moreover, data can be read or written in variable length blocks of from 1 to 8192 words.

These two additions help solve the first problem. The extra core ensures that there is adequate memory space for essential resident code (monitor, filing system, data structure manipulation services, display file output services), for fixed buffers (console control blocks, buffering controller output and input, data structure manipulation area), and for transient code (terminal logon filter, drawing sequencer, library maintenance, text editor, etc.). On the other hand, the disk provides enough storage for a large local library of picture structures and transient code sections. The more transient code sections ("user programs") that are added, the more "intelligent" each terminal appears to the user.

The second problem requires a slightly more complicated solution, involving the design of the terminals themselves and their connection to the PDP-8, rather than the PDP-8 itself. Operations such as real time three-dimensional rotation and clipping currently inflict a large cost penalty on terminal equipment and associated support computers particularly in a multi-terminal environment. These capabilities were thus considered as beyond the range of supportable facilities. The impact of this decision on the choice of terminals was significant. Once the need for a dynamically changing display is eliminated, storage tube CRT terminals become very attractive. No refresh mechanism or storage is required, making these

devices the lowest cost, interactive, full graphics terminals now available. Moreover, there is no upper bound on the amount of data that can be presented on the screen. That is, if a terminal has a refresh memory, and that memory becomes filled, no additional data can be displayed. With a storage tube, this cannot happen.

Of course, there are drawbacks. The first is a 0.5 second nonselective erase time and the second is the lack of display file correspondence for joystick (X, Y) input. The first is really not too annoying since real time, 3D, etc., has been eliminated from the system. The second can be greatly minimized by careful software design.

The use of storage terminals reduces terminal handling to a rather small task, namely just the initial transmission of the picture. However, the mini-computer must be able to service a multiple number of terminals. If one terminal required a complex data structure alteration while another requested a redisplay of a picture, CPU saturation would probably result. (A technical aside is necessary at this point: the picture must be transmitted to the storage terminal one character at a time. This is the result of the differences in display vs. transmission time, as well as the fact that the terminals have no data buffering capability. Thus, the interrupt load to send an initial display is quite high.) To counteract this, a simple timeshared buffering control unit* for initial picture display is needed. This combination of storage terminals plus controller greatly reduces the interrupt load for the PDP-8. Input from a console is merely a few words of joystick (X, Y) coordinate information or a complete text line. Output to a display is usually just a line segment, text string, or subpicture instance, transmitted--as far as the PDP-8 is concerned--in a single block transfer. The controller can easily overlap transmission of characters to as many as eight terminals. With a relatively small number of interrupts to service, the PDP-8 has much more processing time available for use in satisfying data structure operation requests. Hence, the total number of terminals, and/or the quality of support, can be greatly increased.

* While a "simple" device is all that is needed, a PDP-8/I was available for this implementation. It allowed flexibility for testing different types of terminals and different buffering schemes. However, a small hard-wired device would be quite adequate for the buffering needed.

Two other minor but highly useful additions were made to the PDP-8 complex. A sixty cycle clock is used to sequence certain system functions and to keep track of the time of day, while a teletype is used as an operator's console and system log. The operator can specify which terminals are considered operational and can request or reset certain system status flags. The fact that a user has signed on or signed off is printed automatically on the teletype, along with the console number, the user's number, and the time of day. Any illegal sign-on attempts are also noted on the log in a similar manner.

2.2 Remote Section

The remote 360/75 is a typical large scale computing facility. Its resources include 1 megabyte of fast core, 1 megabyte of slow core, 20 online disk drives, 2 drums, several remote entry stations, printers, and card readers. Serving several thousand student, staff, and research users, two to three thousand jobs are run through the facility's batch streams daily. Computational and filing support for the satellite graphics system are obviously within the range of this remote installation's capabilities. The problem, then, is tapping these resources without unduly degrading batch performance. If this can be done efficiently, the occasional need for extensive computational power can be fulfilled inexpensively.

Note that this is a very typical situation. A small group of users with its own hardware sometimes needs the facilities of a large installation that can only be available economically when a large group of users is present. The small group effectively has no influence over the choice of hardware at the remote cite. Moreover, they cannot afford to pay a major share of that installation's costs, so they cannot expect a major share of the available resources. Whereas the satellite hardware was chosen and tailored with the multi-terminal graphics system in mind, in any practical sense, the remote computer must be viewed as a "given" quantity. It exists already; it can be tailored only in the sense that effective software and some careful administrative decisions can extract the needed resources at a reasonable cost to both sets of users.

Resources for archival storage are relatively easy to allocate. Although many variations exist, all remote installations of this size have some charging scheme for allotting online disk and drum space. The vehicle is usually a monthly fee related to the amount of space reserved. Typically, individual users do not acquire these resources on a reserved basis; only the larger subgroups among all users, such as student instructional classes and departmental research groups, actually "rent" online storage space. Hence, some amount of such storage (in the present case, less than half of one disk drive) can be paid for by the graphics system users in the same manner as other subgroups. The cost is reasonable and the method judicial.

Computational power is another matter. This really implies three questions. How long is the job in question in the machine? How much main memory does it tie up while there? What percent of the total available execution time does it use in the period? These items, clock (total) time, memory space, and execution time, have very distinct impacts on the remote system's normal performance. Intended remote support for the satellite system is to be (1) occasional short bursts of number-crunching and (2) occasional archival storage requests. The first will require a very high proportion of total execution time and probably a large amount of memory. The second requires very little execution time and a relatively small amount of memory. The ideal would be to have these remote services available at all times, noting that between infrequent requests of short duration, no resources at all should be in use. Unfortunately, operating systems generally available do not support this. Specifically, when a job (or job-step in the case of OS/360) enters the system, it immobilizes the maximum memory space that it might need at any one instant for its entire sojourn in the machine, even when idle! Even though only a thousand words or less of memory might be needed in the idle state to recognize a request and load the necessary service procedure, 100K words would be continuously unavailable to the installation if that was the job's maximum allocation. Of course, this would be unacceptable to the other users, and rightly so.

The design of the local system is of aid at this point. Significant stand-alone operation including comprehensive data preparation for later use in conjunction with the remote facility is available. Hence, the services of the remote installation need not, in practical terms, be available at all times but only after sufficient material has been prepared locally. Arrangements can be made for the remote services to be available during certain limited time periods. This is, of course, the typical compromise administrative solution for this type of problem. But this compromise would be reached even if no stand-alone ability was present; the small graphics group cannot afford more. The usual graphics system, lacking stand-alone capability, would be "restricted" by the compromise to operate only during the limited hours agreed upon. And part of that time, in turn, would be spent on activities that did not even require the remote facility in the first place. So the satellite system with stand-alone ability has the very distinct advantage of always being available for some form of productive use.

The first two questions, job length and memory requirement, have been skirted. With a reasonable operating system, they could be resolved adequately; in the present case, the problems they pose can only be compromised by administrative decision. The question of execution time is resolved easily: graphics tasks execute at the same priority as batch tasks. Quick operations will be handled rapidly, while long computations will be slowed down by the service requests of other users. However, since no task has abnormally high priority, batch performance is not unduly degraded; the graphics tasks are essentially the same as other normal batch users.

3. SOFTWARE ARCHITECTURE: A USER'S VIEWPOINT

The user interacts with GRASS by means of the joystick and keyboard at his terminal; the display screen of the terminal appears to the user as shown in Figure 2. All data being manipulated, whether simply a page of text or replete with lines and instances, is displayed in the Draw Area. This data is referred to as a "picture" and is represented within the PDP-8 by a "data structure." The remaining nine areas of the screen are used by programs running in the PDP-8 to communicate control information ("frame text") to the user. Information of this type includes: system status messages (System Line Area), option or mode lists (Menu Area), YES/NØ responses (Answer Area), mode termination (Return Area), instructional messages (Prompt Area), function lists (Light Button Area), and ERRØR responses (Panic Area). As appropriate, the particular program being used will alter the messages and options shown on the screen; the terminal user will indicate desired choices with the joystick. In addition, lines entered by the user at the keyboard will appear at the bottom of the screen as they are typed; the line disappears' after being completed (e.g., when sent to the PDP-8 from the buffer control).

To "regenerate" the display means that the screen is first erased, then the frame text is displayed followed by the picture as reconstructed from its data structure in the PDP-8. Note that the data structure for a picture represents two classes of information: generally visible and generally not visible. The first is always displayed by the system automatically during a regeneration; this includes lines, text, and instances of other pictures. The second class is displayed only by the explicit request of a program as initiated by a terminal user's function choice. For example, a picture of an electrical network would be constructed with such things as resistor, capacitor, and transistor instances in the visible class. On the other hand, such information as the node equations for the network and lists of the variables used in the equations would be constructed in the invisible class. Whenever a regeneration is performed, the network, with its resistors, capacitors, and transistors, would appear in the Draw Area. But when the user desired to look at the

network's equations or the list of global variables or the list of local variables, etc., he would use a function command in his program to cause display of the information. The Draw Area might contain, for example, the equations rather than the network image.

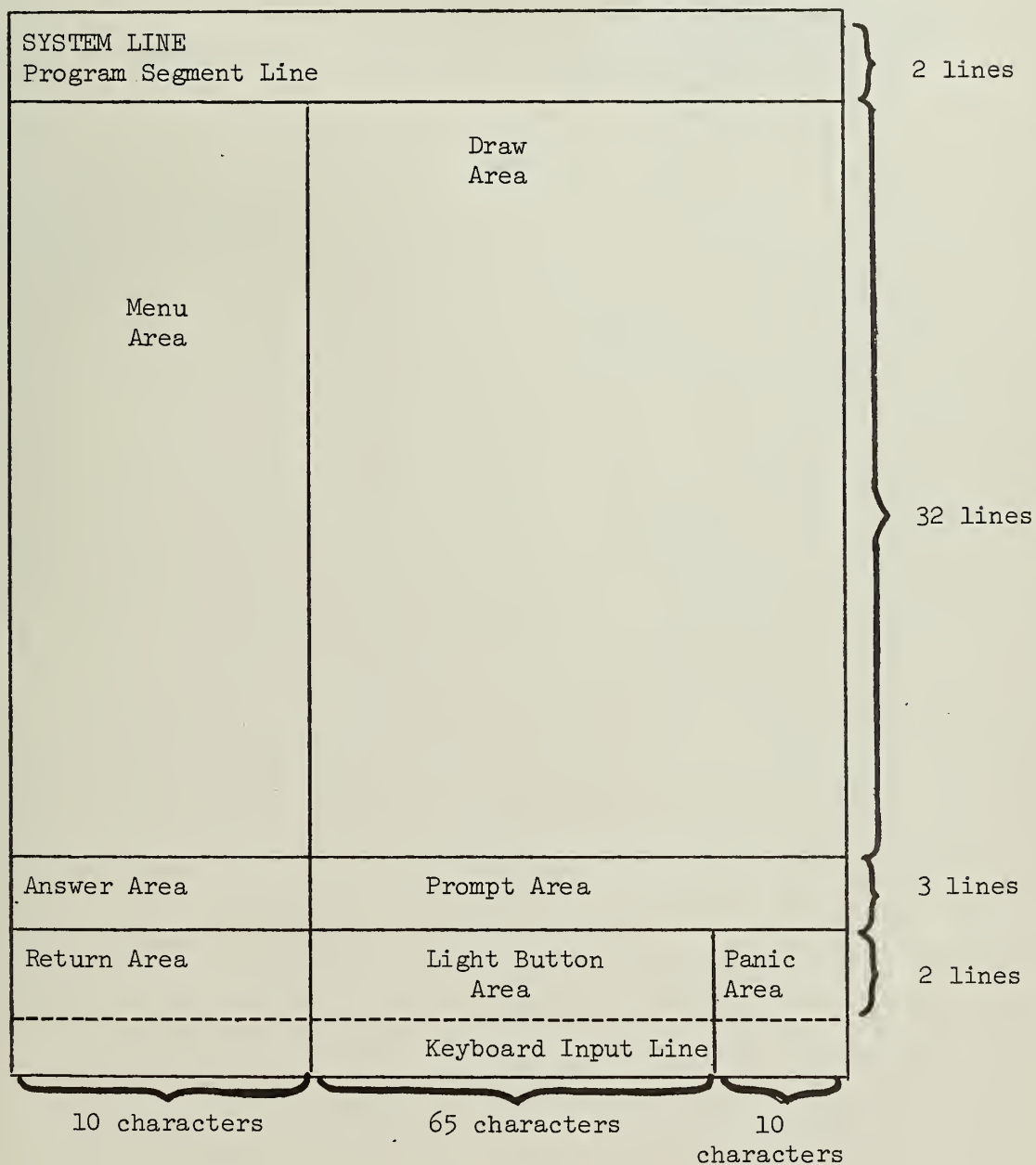
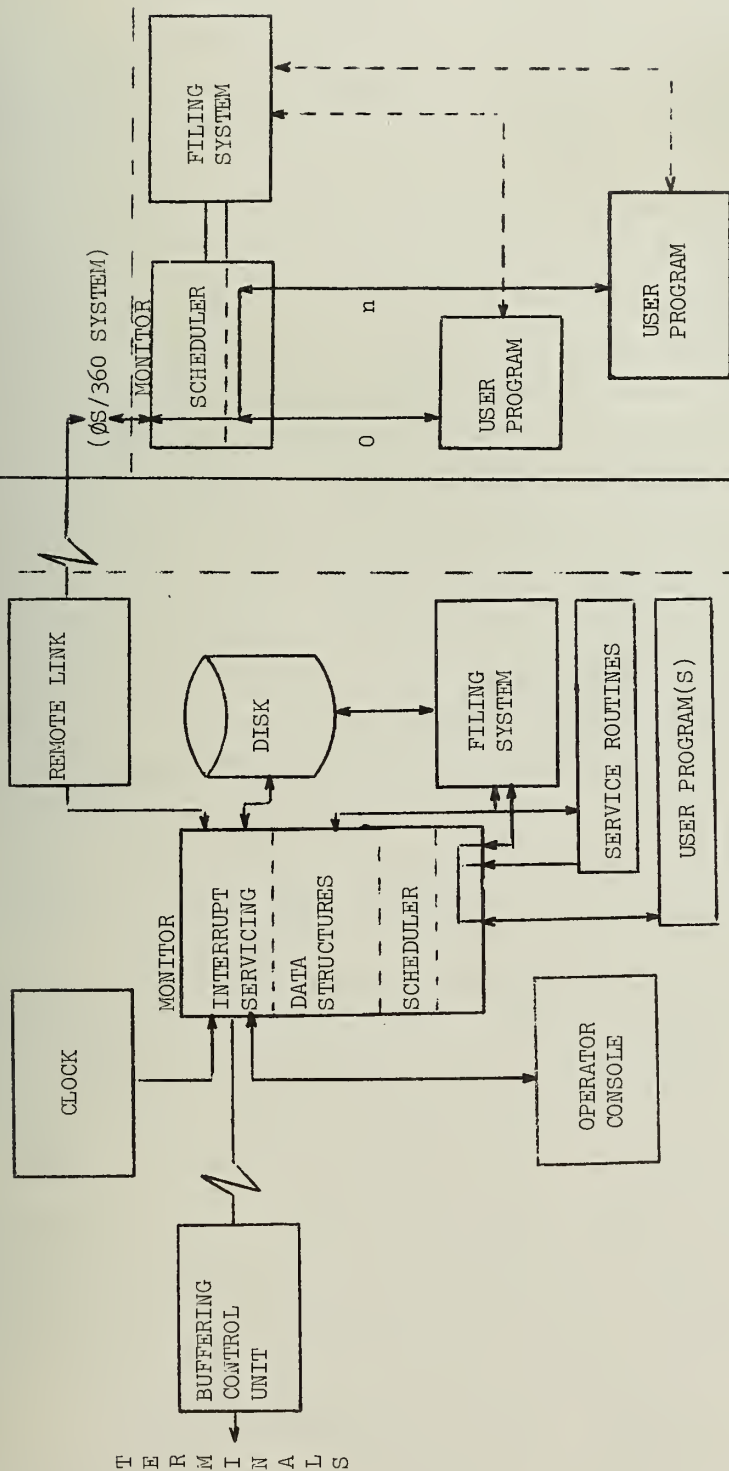


Figure 2. GRASS Display Screen Sectioning Detail

After successfully logging-on to the PDP-8 system, the user may begin execution of any currently available interactive program by choosing it from a list that is displayed in the Menu Area. These programs, which execute entirely in the PDP-8 under a multi-terminal monitor, use system supplied service routines to manipulate the data structure, to access the local filing system, to regenerate the terminal's display, and to transmit data to the remote computer (360/75). Consult Figure 3.



NOTE 1: All code in the PDP-8 is permanently resident except for the User Programs. User Programs are stored on the disk in semi-reentrant code segments. Only one such segment may be in memory at any given time.

NOTE 2: Due to characteristics of ØS/360, all code is resident. However, User Programs can load only needed selected segments of themselves with the ØS/360 XCTL, ILOAD, and LINK macros.

Figure 3. GRASS Software Configuration

Three such programs are CALLER, GNDRW, and REACT. The first is special in that it maintains the list of valid system users and valid available programs; CALLER is the log-on filter automatically invoked by the system when a user begins activity at a terminal. Yet CALLER is just like any user-written program in that it runs under the monitor.

GNDRW and REACT both appear in CALLER's available program list. GNDRW is used to create any arbitrary picture composed of lines, text, and instances of other pictures in the visible data class. Commands are available to store, fetch, and delete these pictures (both visible and invisible parts) from the local filing system. Moreover, other programs can call GNDRW as a subroutine; hence, a program that manipulates a picture in a particular manner or adds additional information to a picture's invisible data class can invoke GNDRW for the initial construction work. REACT is a generalized communications interface for interacting with programs in the remote computer. Functions are available for sending keyboard lines or joystick hits, as well as entire pictures created with GNDRW, to the 360/75. On the other hand, programs running under the multi-terminal monitor in the 360/75 can send text lines or even entire pictures to the PDP-8 for display on the user's terminal or for replacement in the data structure of the terminal's current picture. REACT, as with GNDRW, can be invoked from any other program as a subroutine.

Whenever the 360/75 is available for graphics servicing, the terminal user can send the remote graphics monitor commands via REACT. These commands cause user programs in the 360/75 to be started or halted. Programs running under the remote monitor can use system supplied services to access the remote filing system and to transmit data to the PDP-8. One such program is the remote library maintenance facility, LSD. When directed by requests sent from the terminal user, LSD can store, fetch, and delete pictures in the remote library. Fetching and saving, of course, implies interacting with the appropriate routines in REACT.

A detailed explanation of the operation of a GRASS terminal is given in [7]; facilities available to remote programs are described in [8]. The use of certain remote facilities, as well as the construction of PDP-8 user programs, cannot be initiated without a detailed understanding of some of the system's inner workings. This information is provided in [9, 10].

Use of the system is exemplified by the Simulation and Modeling (SAM) packages described in [5]. This application is intended to numerically solve physical network problems input by the terminal user. Part of the package runs as a user program in the PDP-8, GSAMV1; the rest runs as user programs in the 360/75. After logging-on, the terminal user chooses GSAMV1 from the available program list displayed by CALLER, and execution begins. GSAMV1 then displays its own list of three available functions.

The first function merely calls GNDRW as a subroutine; this allows construction of the visible portion of a picture. Note that instances (resistors, etc.) in GNDRW are representations of previously constructed pictures (networks); a user can thus construct and use whatever "primitives" (instances) he desires. The instance "transistor" represents a network that models a transistor using resistors and capacitors as instances. The instance "resistor" represents a network that models a resistor; this model does not contain any instances (it is primitive). The model of the resistor was itself constructed by the user with GNDRW.

The second function allows the user to specify equations, variables, and parameters to be associated with the network. This consists of calling a simple text editor, which happens to be a sub-function of GNDRW, and typing in the appropriate information. Data created via this second function is added to the invisible portion of the picture.

The final function is merely a call to REACT; if the 360/75 is available, the entire network just created can be transmitted and stored remotely via LSD for later analysis. On the other hand, the network could be transmitted to the first phase of the SAM package instead of to LSD; analysis could then begin immediately. An interactive process results in which intermediate output is transmitted from the SAM packages to the terminal, and new variable and parameter values are transmitted back from the user. When analysis is complete, the user can have SAM read a previously saved network from the remote filing system and begin a

new analysis. Or another new network can be transmitted from the PDP-8. Or the user could terminate SAM in the 360/75 and return to the first function to create another network.

The above application points out certain salient features of the system. GNDRW is general enough to handle the visible portion of any 2D image required. After all, a text page is a degenerate network: no instances, no lines. A molecule or a highway bridge or the economy of Rhode Island are all representable as networks: they just use differently drawn instances. The functioning of the instances and the meaning of the entire picture depends on the information that is associated with the visible image by the user. This extra information is easily added by GSAMV1 or by any other similar (and simple) user application program. Obviously, applications that are just visual in nature or not intended as input for analysis (architectural or mechanical drawing, page layouts, manuscript creation, etc.) do not even need extra descriptive information. Such applications can be largely supported by GNDRW alone. The range of applications for GRASS, both stand-alone and with remote support, is thus extremely large. The only limit, in fact, is the imagination of the users.

4. OPERATIONAL EXPERIENCE

The entire system has been operational, although with only three of the possible maximum of eight terminals, since the first quarter of 1971. If a completely new display of a fairly complex picture* is requested from all three terminals simultaneously, total elapsed time for the three displays is from 4 to 6 seconds, depending on the pictures involved. This is considered the upper bound for average worst-case response. Under normal production use with eight terminals, taking into account the speed at which users typically work, simultaneous requests from three terminals is very unlikely.

Total time on a single terminal for a new display of a similarly complex picture is only 2 to 3 seconds. Of course, the most often performed operation, by far, is not a new display, but rather the simple addition of a vector, text line, or subpicture instance to an existing display. This is performed very quickly: typically 50 to 100msec. for a vector or text line, 100 to 250msec. for an instance. All of these response times seem quite satisfactory.

Response from the remote computer has also been satisfactory on the average. However, any specific request may be delayed if batch or keyboard terminal requests are particularly heavy at the same time. This is due mainly to the fact that graphics support runs at the same priority level as batch and at lower priority than keyboard terminals. Even in this mode of operation, 95 percent of all command or filing system requests are handled within 500ms. Requests requiring extensive computation naturally need a longer period to run to completion. For example, analyzing a simple network of less than ten elements may take from 500ms to three seconds. Much larger networks may require up to several minutes. In any event, requests originating from the graphics terminals receive the same servicing as any other requests; the remote installation does not degrade its regular service to other users. And, response at the graphics terminal is quite reasonable for the services requested.

* Fifteen to thirty subpicture instances plus some text and lines as well as the frame and all frame text areas.

The combination of limited remote support plus extensive stand-alone ability has worked out very efficiently. Intense preparation of manuscripts, diagrams, or networks can be carried out at all times, regardless of the condition of the remote computer. Whenever the remote installation is available for graphics support, completed items are transmitted for inclusion in the remote filing system or for analysis. Remote resources are not squandered on data preparation.

5. EXTENSIONS

Hardware that has recently become available could be used at practically no cost increase to improve the performance of the satellite system. In place of the original PDP-8, any of several new 16-bit mini-computers could be used. Various features of these machines including extended arithmetic, larger instruction repertoires, etc., would improve the efficiency of the software; hence, improve response. Features available to users would also be improved. A minor example of this is the character set. To conserve space, characters are packed two per memory word; on a 12-bit machine this means only 64 characters are available. With a 16-bit machine a full character set could be utilized. Other examples would be the addition of support for conic sections and other curves as well as 2D rotations.

Certain other improvements can be made as technologically improved, lower priced terminals become available. Over the next few years reasonably priced high-speed memory should become available. This would mean that a refreshed type terminal could replace the storage units. With very minor software changes, a significant response improvement can be expected. First, the data for a picture would be transmitted to the terminal in a block, rather than character by character. Speed is increased and the buffering controller is eliminated. The 0.5 second needed as erase time is also eliminated. Second, joystick (X, Y) input would now have a display file correspondence, thus making more rapid identification of entities possible. Additionally, selective removal of data from the display memory would be available, thus reducing the necessity for complete picture regeneration.

These hardware improvements lead, of course, to reconsideration of the range of the 2D system. If response can be improved as indicated above, perhaps some features that were previously left out (such as real time rotation, clipping, and 3D) can now be included. Unfortunately, these features, unless aided with terminal hardware, still require massive number-crunching just for one terminal, let alone for several. The newest

mini-computers available now, or expected, obviously do not have this capability. On the other hand, terminal hardware of the sort needed is still rather esoteric and appropriately expensive. Adequate cost improvements do not seem likely, even with ISI, for the next several years. Hence, a timeshared, mini-computer configuration still seems to be the most powerful that will be available, economically, in the near future. However, response and selected features should show some marked advances.

Improvement at the remote computer end lies mainly in increased user program convenience software. As discussed earlier, no control can be exercised over the hardware or operating system of the remote cite. Obviously, if the computer is replaced with a more powerful model or if a new release of the operating system is more efficient than the old, increased response will be seen by all users.

6. CONCLUSIONS

A reasonably powerful, relatively low cost, timeshared graphics facility has been successfully implemented. This facility can be easily upgraded using more recent vintage hardware. A system configuration such as that presented apparently will remain economically and technologically viable for the next several years. Reproduction of the system can be easily accomplished with standard off-the-shelf hardware (c.f. Appendix).

The resources of a powerful remote computing facility were made available to the satellite system without unduly degrading the remote system's normal performance. Extensive but low cost computational and archival storage capabilities were thus provided.

LIST OF REFERENCES

- [1] Quarterly Progress Reports, Department of Computer Science Section 3.3, University of Illinois, Urbana, Illinois 61801, March 1970-June 1971.
- [2] Gear, C. W. An Interactive Graphic Modeling System, Department of Computer Science Report No. 318, University of Illinois, Urbana, Illinois 61801, April 1969.
- [3] Gear, C. W. Generalized Simulation and Modeling System, Department of Computer Science File No. 785, University of Illinois, Urbana, Illinois 61801, December 1968.
- [4] Michel, M. J. GRAPHICAL REMOTE-ACCESS SIMULATION SYSTEM (GRASS) The Communication and Monitor Components (GASP/GLASP), Department of Computer Science Report No. 346, University of Illinois, Urbana, Illinois 61801, August 1969.
- [5] Gear, C. W., et.al. The Simulation and Modeling System--A Snapshot View, Department of Computer Science File No. 824, University of Illinois, Urbana, Illinois 61801, February 1970.
- [6] Levin, H. Local Information Retrieval for the Simulation and Modeling System, Department of Computer Science Report No. 409, University of Illinois, Urbana, Illinois 61801, August 1970.
- [7] Michel, M. J. and Koch, J. GRASS: Terminal User's Guide, Department of Computer Science Report No. 467, University of Illinois, Urbana, Illinois 61801, August 1971.
- [8] Haskin, R., Nickolls, J., and Michel, J. J. GRASS: Remote Facilities Guide, Department of Computer Science Report No. 466, University of Illinois, Urbana, Illinois 61801, July 1971.
- [9] Michel, M. J. and Haskin, R. GRASS: Extended Remote Facilities Guide, Department of Computer Science File No. 867, University of Illinois, Urbana, Illinois 61801, August 1971.
- [10] Michel, M. J. GRASS: System Software Description, Department of Computer Science Report No. 468, University of Illinois, Urbana, Illinois 61801, August 1971.
- [11] Haskin, R. GRAPHICS 8: System Maintenance (Software), Department of Computer Science File No. 865, University of Illinois, Urbana, Illinois 61801, July 1971.
- [12] Nickolls, J. GRAPHICS 8: System Maintenance (Hardware), Department of Computer Science File No. 866, University of Illinois, Urbana, Illinois 61801, August 1971.

- [13] Carter, C. E., et.al. GRAPHICS 8: Graphics Hardware--1469 Gear, Department of Computer Science Report No. 371, University of Illinois, Urbana, Illinois 61801, December 1969.
- [14] Carter, C. E., et.al. DEC PDP-8 Interface to IBM 2701 PDA, Department of Computer Science Report No. 372, University of Illinois, Urbana, Illinois 61801, March 1970.
- [15] Lopeman, H. E. GRAPHICS 8: PDP-8 Interface Hardware--1469 Gear, Department of Computer Science File NO. 828, University of Illinois, Urbana, Illinois 61801, March 1970.
- [16] Levin, H. and Lopeman, H. PDP-8/I to PDP-8 Interface, Department of Computer Science File No. 843, University of Illinois, Urbana, Illinois 61801, June 1970.
- [17] Hyde, C. PDP-8 Disk Interface, Department of Computer Science File No. 861, University of Illinois, Urbana, Illinois 61801, November 1970.
- [18] COMPUTEK User's Manual, Series 400/15 CRT Display System, Computek Inc., Bulletin 400M, July 1969.

APPENDIX

To establish a ball park idea of the costs involved, the system could easily be reconstructed from the following list of sample equipment available now, off-the-shelf. Emphasized is the fact that these costs are upper-bound for off-the-shelf items. That this list includes several DEC items is NOT essential, as others are available, though not as well known. Items with an asterisk are not essential but add some convenience to the complex.

1. Mini-Computer Complex

Processor:

PDP-8 computer	\$10K
3 extra core modules	15K
RF08 disk	<u>15K</u>
	\$40K

Buffering Controller:

PDP-8/I computer	\$10K
1 extra core module	<u>5K</u>
	\$15K

Other Hardware (Minor):

Terminal multiplexer, Clock, etc.	\$ 6K
*hardcopy output device for terminal use	4K
*magnetic tape unit for system IPL	<u>5K</u>
	\$15K

2. Software

conversion (since versions already exist)	\$10K
--	-------

3. Terminals

8, 10K @	\$80K
----------	-------

TOTAL: \$160K for eight intelligent graphics terminals in a complete, stand-alone, timeshared environment.

The reader is encouraged to look up the prices of other hardware currently available. An example, though not a particularly good one, is the IBM 2250 Model 1 display unit. This terminal is refreshed from its own 8K-byte core memory, and is fast enough for some, but not a lot of, real time, 3D, rotational work. The cost of just the terminal is about \$100K! And the CPU needed to drive just one of these is comparably expensive.

U. S. ATOMIC ENERGY COMMISSION
UNIVERSITY-TYPE CONTRACTOR'S RECOMMENDATION FOR
DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

AEC REPORT NO.

C00-1469-0187

2. TITLE

GRASS: System Overview

TYPE OF DOCUMENT (Check one):

☒ a. Scientific and technical report

☐ b. Conference paper not to be published in a journal:

Title of conference _____

Date of conference _____

Exact location of conference _____

Sponsoring organization _____

☐ c. Other (Specify) _____

RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

☒ a. AEC's normal announcement and distribution procedures may be followed.

☐ b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.

☐ c. Make no announcement or distribution.

REASON FOR RECOMMENDED RESTRICTIONS:

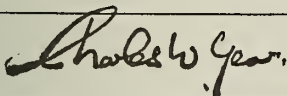
SUBMITTED BY: NAME AND POSITION (Please print or type)

C. W. Gear, Professor
and Principal Investigator

Organization

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

Signature



Date

July 1971

FOR AEC USE ONLY

AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION
RECOMMENDATION:

PATENT CLEARANCE:

☐ a. AEC patent clearance has been granted by responsible AEC patent group.

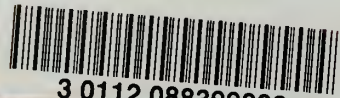
☐ b. Report has been sent to responsible AEC patent group for clearance.

☐ c. Patent clearance not required.

SEP 28 1971



UNIVERSITY OF ILLINOIS-URBANA
510.84 IL6R no. C002 no.463-468(1971)
Control point design using modular logic



3 0112 088399883